CS 270: Combinatorial Algorithms and Data Structures Spring 2023

Lecture 18 — March 16, 2023

Prof. Jelani Nelson *Scribe: Nate Tausik*

1 Overview

In the last lecture we introduced primal/dual analysis for approximation algorithms (in particular, for weighted set cover). In this lecture we will give another example of primal/dual analysis for approximation algorithms, then look at barriers to using LPs for improving approximation ratios. Finally, we will introduce the concept of PTAS and FPTAS.

2 Vertex Cover with Primal/Dual Analysis

2.1 Vertex Cover

In vertex cover we are given an undirected graph $G = (V, E)$, and we want to pick a subset $S \subseteq V$ of vertices that cover all edges in E, i.e. for each edge in E, at least one of its endpoints is in S. Note that vertex cover can be viewed as a special case of set cover where the universe is E, and each vertex v corresponds to the subset consisting of all edges v touches.

We start by giving a greedy algorithm for solving vertex cover.

Algorithm 1 Greedy Vertex Cover

```
$S \leftarrow \emptyset$
while $\exists e = (u, v) \in E$ not covered do
    $S \leftarrow S \cup \{u, v\}$
```

Claim 2.1. Using the above algorithm, $|S| \leq 2 \cdot \mathrm{OPT}$.

For a direct proof of this claim see CS 170. We want to move towards a primal/dual analysis of this algorithm. First, however, we hint at some open problems.

Conjecture 2.2. *No polynomial time algorithm for vertex cover does better than a 2-approximation.*

We do not explore the details of this conjecture. However, if it were false, it would imply that the **Unique Games Conjecture** is false. The Unique Games Conjecture was proposed by Subhash Khot in 2002, and claims that a certain problem called Unique Games is NP-Hard [1]. More, broadly, if we had P=NP, we could find an exact polynomial time solution to vertex cover.

2.2 Primal/Dual Analysis

We will now set up the primal LP for vertex cover. As before, we can describe an integer program whose feasible values exactly correspond to possible solutions of vertex cover. Then, we can make a continuous version which simply drops the requirement that the variables are integers. The continuous version is known as an **LP relaxation**. We give an LP relaxation for vertex cover.

$$\min \sum_{v \in V} x_v \quad \text{such that} \quad x_u + x_v \geq 1 \quad \forall e = (u, v) \in E, \quad \mathbf{x} \geq \mathbf{0}$$

We can dualize the LP as in last lecture. As always, we get one dual variable for each primal constraint, and one dual constraint for each primal variable.

$$\max \sum_{e \in E} y_e \quad \text{such that} \quad \sum_{e \in E \,:\, e \text{ touches } v} y_e \leq 1 \quad \forall v \in V, \quad \mathbf{y} \geq \mathbf{0}$$

Before using the LPs to prove Claim 2.1, we note that there is another simple 2-approximation algorithm.

Remark 2.3. Consider solving the LP relaxation. For each edge $e = (u, v)$, one of x_u or x_v must be at least 0.5, since otherwise the solution is not feasible. So, if we round x_v to the nearest integer for all $v \in V$ (rounding 0.5 up), we get a feasible solution which is at most twice the cost of the optimal cost of the LP relaxation. The optimal cost of the LP relaxation is at most OPT for vertex cover, so we have a 2-approximation for vertex cover.

Now, we finally perform the primal/dual analysis, giving a proof of Claim 2.1.

Proof of Claim 2.1. We run the greedy Algorithm 1, and consider how the variables in our primal and dual LPs update to reflect the progress of the algorithm.

- Initially we have no vertices in our cover, so set $\mathbf{x}, \mathbf{y} \leftarrow \mathbf{0}$
- Whenever $e = (u, v)$ causes us to add its vertices to S, we set $x_u \leftarrow 1, x_v \leftarrow 1, y_e \leftarrow 1$

So, based on the update rules, we see that the cost(Primal)= 2·cost(Dual) at each step. If our solutions $\mathbf{x}$ and $\mathbf{y}$ are feasible, then by the same argument as last time, we have

$$\text{cost(Dual)} \leq \text{OPT} \leq \text{cost(Primal)} = 2 \cdot \text{cost(Dual)} \leq 2 \cdot \text{OPT}$$

and we are done. The primal is feasible because by the time the algorithm finishes, we have covered all uncovered edges. Also, the dual is feasible because once $y_e = 1$ for some $e = (u, v)$, both u and v are in the cover, so we will not get $y_f = 1$ for any other edge f which touches u or v. So, we are done. □

3 Integrality Gaps

Now, we will apply the primal/dual reasoning to find barriers to improving approximation ratios via LP. In particular, we will find lower bounds on the approximation ratios that LP relaxations can give us for vertex cover and set cover. The main insight is to notice that in our primal/dual proof above, OPT was actually OPT of the LP relaxation while what we really want is OPT of the integer program (which is the same as the true OPT of our original problem). If there is a gap between the OPT of the IP and the OPT of the LP relaxation, this puts a bound on how good our approximation from the LP relaxation can be. This idea naturally leads us to our key definition.

Definition 3.1. The integrality gap is

$$\frac{\text{worst case over all problem instances for OPT of the } \textbf{IP}}{\text{worst case over all problem instances for OPT of the } \textbf{relaxed LP}}$$

The integrality gap is the barrier to improving approximation ratios with the relaxed LP.

3.1 Vertex Cover

In this section, we will show that the integrality gap for vertex cover is at least ≈ 2, where the relaxed LP is as given above. Since we already gave a 2-approximation algorithm, we know the integrality gap has an upper bound of ≈ 2. Thus, combining these facts, we may conclude that the integrality gap is ≈ 2. To show the lower bound, we must find an example of vertex cover where the ratio of cost of IP to cost of relaxed LP is ≈ 2.

Claim 3.2. The integrality gap for vertex cover with the relaxed LP as above is at least ≈ 2.

Proof. Consider $G = K_n$, the complete graph on n vertices. We claim OPT of the IP is $n-1$. It cannot be more than $n-1$ since every edge is connected to two distinct vertices, and so certainly taking all but one vertex suffices. If we could cover the graph with fewer vertices, there would be some pair of vertices $\{u, v\}$ not in the cover. Since G is complete, there is an edge (u, v) which is not being covered, a contradiction. So, OPT of the IP is $n-1$. By taking $\mathbf{x} = (\frac{1}{2}, \frac{1}{2}, \dots, \frac{1}{2})$, we see that OPT of the relaxed LP is at most $\frac{n}{2}$. So, the integrality gap is at least $\frac{n-1}{n/2} \approx 2$. $\square$

It is worth noting that this argument, which shows the best LP-based approximation algorithm gives 2·OPT, follows the typical pattern. The pattern is to prove the lower bound via an example (Claim 3.2) and to prove the upper bound via an algorithm (Claim 2.1). Before moving on, we make a tangential remark with an important point.

Remark 3.3. The Sherali-Adams hierarchy gives a pocedure for adding constraints to a relaxed LP to make it more like the IP it came from. In fact, with enough constraints, one can fully capture the original IP. However, one would need exponentially many constraints which renders the relaxed LP useless. The important takeaway is the LP relaxation is not unique; our bounds (for instance, those in the section above) are dependent on the specific relaxtion.

As another tangential remark in a different direction, we will see in a future lecture that we can improve on some results using semidefinite programming, or other schemes that are more flexible than linear programming. For now, however, we continue with linear programming.

3.2 Set Cover

Recall the relaxed LP for set cover from last time, copied below.

$$\min \sum_{S \in \mathcal{C}} x_S \quad \text{such that} \quad \sum_{S \ni i} x_S \geq 1 \quad \forall i \in [n], \quad \mathbf{x} \geq \mathbf{0}$$

As in the last section, we will give an example to put a lower bound on the integrality gap for set cover with the above relaxed LP.

For some $q > 0$, put $U = \mathbb{F}_2^q \setminus \{\mathbf{0}\}$, the nonzero q-dimensional vectors mod 2. Notice $|U| = 2^q - 1$. Put $\mathcal{C}$ as the collection of subsets $S_v = \{\alpha \in U \mid \langle \alpha, v \rangle \equiv 1 \pmod 2\}$ for each $v \in \mathbb{F}_2^q$. Notice $|\mathcal{C}| = 2^q$. Define $m = 2^q$.

Claim 3.4. For all $\alpha \in U$, there are $\frac{m}{2}$ values of $v \in \mathbb{F}_2^q$ which satisfy $\alpha \in S_v$.

Proof. We use probability. Pick $v \in \mathbb{F}_2^q$ uniformly at random, and consider $\mathbb{P}(\langle \alpha, v \rangle \equiv 1 \pmod 2)$. Each component of α is 0 or 1. Say the first 1 is in position j^*, so $\alpha = (0, 0, \ldots, 0, 1, *, *, \ldots, *)$. We have

$$\langle \alpha, v \rangle = \sum \alpha_i v_i = v_{j^*} + \sum_{k > j^*} * \cdot \text{blah}$$

The sum will be 0 or 1 mod 2. Since v is random, v_{j^*} will be 0 or 1 with equal probability. Since we consider the inner product mod 2, v_{j^*} will flip the value of the sum with probability $\frac{1}{2}$. So, $\mathbb{P}(\langle \alpha, v \rangle \equiv 1 \pmod 2) = \frac{1}{2}$. The v for which we get 1 are exactly those in S_v, and so since we pick v uniformly,

$$\mathbb{P}(\langle \alpha, v \rangle \equiv 1 \pmod 2) = \frac{|S_v|}{m}$$

Combining our two results, $\frac{1}{2} = \frac{|S_v|}{m}$, so $|S_v| = \frac{m}{2}$. □

Since each $\alpha \in U$ is in exactly $\frac{m}{2}$ of our subsets, we see by inspecting the relaxed LP that $\mathbf{x} = (\frac{2}{m}, \frac{2}{m}, \ldots, \frac{2}{m})$ is a feasible solution. So, looking at the objective function, OPT of the relaxed LP is at most 2. Now, we turn our attention to the IP.

Claim 3.5. OPT of the IP is at least q.

Proof. Suppose not, so we can find $S_{v_1}, \ldots, S_{v_{q-1}}$ which are feasible. Since they cover U, we must have $\bigcap_{i=1}^{q-1} \bar{S_{v_i}} = \emptyset$, where the complement $\bar{S_{v_i}} = \{\alpha \in U : \langle \alpha, v \rangle \equiv 0 \pmod 2\}$. We notice intersecting $\mathbb{F}_2^q$ with $\bar{S_{v_i}}$ is the same as imposing a linear constraint, so the resulting intersection is a $(q-1)$-dimensional subspace. Each additional $\bar{S_{v_j}}$ which we add to the intersection will give us an additional linear constraint, and reduce the dimension of our subspace by 1 if it is a new constraint. We only have $q-1$ such $\bar{S_{v_i}}$'s, so the resulting intersection will be at least $q-(q-1) = 1$-dimensional. Our universe U excludes $\mathbf{0} \in \mathbb{F}_2^q$, but even so, a ≥ 1-dimensional subspace must have a nonzero vector, so $\bigcap_{i=1}^{q-1} \bar{S_{v_i}} \neq \emptyset$, a contradiction! □

Now, $q \approx \log_2 |U| = \log_2 n$, and so we have an integrality gap of at least $\approx \frac{\log_2 n}{2}$. We got a $\ln n$ upper bound on the gap using an algorithm from last time, so we have considerably narrowed the possible range of approximations with this relaxed LP.

Remark 3.6. There are more sophisticated examples to bring the lower bound closer to $\ln n$. Uriel Feige showed that if set cover can be approximated to better than $(1 - o(1)) \ln n$ in polynomial time, then SAT can be solved in $O(n^{\lg \lg n})$ [2].

4 PTAS and FPTAS

In the final portion of the lecture, we begin to examine PTAS and FTPAS, which are more flexible types of approximation algorithms.

Definition 4.1. A Polynomial Time Approximation Scheme (PTAS) is a family of algorithms indexed by ϵ such that

1. $\mathcal{A}_\epsilon$ is an approximation algorithm getting at most $(1+\epsilon)\cdot$OPT.

2. The runtime of $\mathcal{A}_\epsilon$ is $O(n^{f(1/\epsilon)})$.

Notice that the runtime of a PTAS can get very bad as ϵ becomes small, depending on what f is. We give a stronger definition.

Definition 4.2. A Fully Polynomial Time Approximation Scheme (FPTAS) is a PTAS where $\mathcal{A}_\epsilon$ has runtime at worst $\text{poly}(\frac{n}{\epsilon})$.

Remark 4.3. PTAS and FPTAS seem to be much more useful in the real world than generic approximation algorithms. If you're proposing new bus routes, it would be much better to say they are within 1% of the optimal, not within a factor of 2 of the optimal, even if the time to find such a route may be slower.

Remark 4.4. We can immediately rule out the existence of an FPTAS for some problems just by assuming P$\neq$NP. If vertex cover had a FPTAS, take $\epsilon = \frac{1}{3n}$. OPT cannot exceed n, so we get within 1/3 of OPT. But the true solution will be an integer, so we really must have the OPT solution. Thus, we solved vertex cover in polynomial time, contradicting P$\neq$NP.

We now start on an extended example of (F)PTAS, which will not be finished until next lecture.

4.1 Knapsack

Recall the knapsack problem. Our input is n items, each with a value and weight pair (v_i, w_i), and a knapsack which can hold W pounds. Our goal is to maximize the total value subject to the weight limit. We can set up an integer program.

$$\max \sum_{i=1}^{n} x_i v_i \quad \text{such that} \quad \sum_{i=1}^{n} x_i w_i \leq W, \quad \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}, \quad \mathbf{x} \in \mathbb{Z}^n$$

There are a couple of exact algorithms for knapsack. The first is a familiar dynamic programming algorithm from CS 170.

- Put $f(i, w) =$ max value attainable with capacity w knapsack, only taking from first i items.
- Recursively define $f(i,w) = \begin{cases} -\infty, & w < 0 \\ 0, & i = 0 \\ \max\{f(i-1,w), v_i + f(i-1, w-w_i)\}, & \text{otherwise} \end{cases}$
- Starting from base cases, calculate $f(n, W)$ with memoization.

Each calculation of $f(i, w)$ takes constant time, so this algorithm takes $O(nW)$ time overall. We also sketch another dynamic programming algorithm.

- Put $f(i, v) =$ min weight needed to take total value exactly v from the first i items.
- Set up a recurrence for $f(i, v)$ and compute all values up to $f(n, V)$ where $V = \sum v_i$.
- Find the max v such that $f(n, v) \leq W$.

The recurrence will again be such that each $f(i, v)$ takes $O(1)$ time, and finding the max takes $O(V)$ time. So, we take $O(nV)$ time overall. So, we have two great solutions for knapsack! Right?

Remark 4.5. Both of the algorithms above are actually very slow. The space complexity of the input weights and values will be $O(\lg W)$ and $O(\lg V)$, so our algorithms are really exponential in the input sizes. These types of algorithms are called **pseudopolynomial**.

Can we find a fast 2-approximation? As usual, we start by removing the $\mathbf{x} \in \mathbb{Z}^n$ constraint from our IP above to form a relaxed LP. We claim without proof that OPT for the relaxed LP is given by greedily picking items based on maximum $\frac{v_i}{w_i}$ ratio. How does this strategy do for the IP? We claim it does very badly.

Example 4.6. Consider a knapsack problem with 2 items, $v_1 = 1 + \delta$, $w_1 = 1$, and $v_2 = W$, $w_2 = W$. Suppose $1 + \delta \ll W$. The greedy strategy will take item 1 first since its ratio is better, and then can take nothing else. However, we could have gotten $W \gg 1 + \delta$ value if we took item 2. So, we actually have at best a W-approximation, which we could have gotten by just taking the highest value item and nothing else (assuming weights are in $\mathbb{Z}$).

However, this silly comparison does give us an idea for another algorithm.

- Try running the greedy ratio-based algorithm.
- Try just taking the most valuable item and nothing else.
- Do whichever strategy gives more value.

Claim 4.7. The above strategy gives us at least $\frac{1}{2}$·OPT.

We will start by proving this claim next time.

References

[1] Subhash Khot. On the Power of Unique 2-Prover 1-Round Games. *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, 767–775, 2002.

[2] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.